Sauteed Onions

Transparent Associations from Domain Names to Onion Addresses

Rasmus Dahlberg
rasmus.dahlberg@kau.se
Karlstad University
Karlstad, Sweden

Paul Syverson
paul.syverson@nrl.navy.mil
U.S. Naval Research Laboratory
Washington D.C., USA

Linus Nordberg
linus@verkligendata.se
Verkligen Data AB
Stockholm, Sweden

Matthew Finkel
sysrqb@torproject.org
Independent
Grand Junction CO, USA

ABSTRACT

Onion addresses offer valuable features such as lookup and routing security, self-authenticated connections, and censorship resistance. Therefore, many websites are also available as onionsites in Tor. The way registered domains and onion addresses are associated is however a weak link. We introduce *sauteed onions, transparent associations from domain names to onion addresses*. Our approach relies on TLS certificates to establish onion associations. It is much like today's onion location which relies on Certificate Authorities (CAs) due to its HTTPS requirement, but has the added benefit of becoming public for everyone to see in Certificate Transparency (CT) logs. We propose and prototype two uses of sauteed onions: certificate-based onion location and search engines that use CT logs as the underlying database. The achieved goals are *consistency of available onion associations*, which mitigates attacks where users are partitioned depending on which onion addresses they are given, *forward censorship-resistance* after a TLS site has been configured once, and *improved third-party discovery of onion associations*, which requires less trust while easily scaling to all onionsites that opt-in.

CCS CONCEPTS

• **Security and privacy** → *Browser security*; **Web protocol security**; • **Networks** → **Naming and addressing**; • **Social and professional topics** → **Technology and censorship**.

KEYWORDS

Onion Services, Web PKI, TLS Certificates, Certificate Transparency

ACM Reference Format:
Rasmus Dahlberg, Paul Syverson, Linus Nordberg, and Matthew Finkel. 2022. Sauteed Onions: Transparent Associations from Domain Names to Onion Addresses. In *Proceedings of the 21st Workshop on Privacy in the Electronic Society (WPES '22), November 7, 2022, Los Angeles, CA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3559613.3563208

WPES '22, November 7, 2022, Los Angeles, CA, USA
© 2022 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9873-2/22/11.
https://doi.org/10.1145/3559613.3563208

1 INTRODUCTION

Onion addresses are domain names with many useful properties. For example, an onion address is self-authenticated due to encoding its own public key. It also makes integral use of the anonymity network Tor to provide secure and private lookups as well as routing [12]. A major usability concern is that onion addresses are random-looking strings; they are difficult to discover, update, and remember [49]. Existing solutions approach these limitations in different ways, e.g., ranging from setting onion addresses in HTTP headers over HTTPS with so-called *onion location* [36] and bookmarking found results to making use of manually curated third-party lists [29, 37, 47] as well as search engines like DuckDuckGo or `ahmia.fi` [34, 49].

Herein we refer to the unidirectional association from a domain name to an onion address as an *onion association*. The overall goal is to facilitate transparent discovery of onion associations. To achieve this we rely on the observation that today's onion location can be implemented in certificates issued by Certificate Authorities (CAs). This is not an additional dependency because onion location already requires HTTPS [36]. The main benefit of transitioning from HTTP headers to TLS certificates is that all such onion associations become signed and sequenced in tamper-evident Certificate Transparency (CT) logs [21, 23], further tightening the relation between CAs and onion keys [5, 6, 45] as well as public CT logging and Tor [10, 29].

Our first contribution is to make onion associations identical for all Tor users, and otherwise the possibility of inconsistencies becomes public via CT. Consistency of available onion associations mitigates the threat of users being partitioned without anyone noticing into subsets according to which onion address they received during onion association. Our second contribution is to construct a search engine that allows Tor users to look up onion associations without having to trust the service provider completely. Other than being helpful to validate onion addresses as authentic [49], such discovery can continue to work *after* a TLS site becomes censored.

Section 2 briefly covers CT preliminaries. Section 3 describes *sauteed onions*, an approach that makes discovery of onion associations more transparent and censorship-resistant compared to today. Section 4 discusses related work. Section 5 concludes the paper. Appendix A contains query examples for our search engine. Appendix B outlines an example setup. All artifacts are online [2].

2 CERTIFICATE LOGGING PRELIMINARIES

CT is a system of public append-only logs that store TLS certificates issued by trusted CAs [21, 23]. If web browsers add the criterion that a certificate must be logged before accepting it as valid, certificate issuance practices by CAs effectively become transparent so that mistakes and malfeasance can be detected by anyone that observes the logs. These observers are called *monitors* because they download every certificate from all logs. One can self-host a monitor, or use a third-party service like `crt.sh`, or follow other models based on subscriptions [8, 24]. To avoid introducing more parties that are trusted blindly as in the CA ecosystem, CT stands on a cryptographic foundation that permits efficient verification of inclusion (a certificate is in the log) and the append-only property (no certificate has been removed or modified) [13]. A party engaging in verification of these (logarithmic) proofs is called an *auditor*.

In practice, CT has been rolled-out gradually to not break the web [42]. One facilitating factor has been the introduction of Signed Certificate Timestamps (SCTs). An SCT is a log's *promise* to include a certificate within a certain amount of time; typically 24 hours. This guarantees low-latency certificate issuance so that CAs can *embed* SCTs in certificates to keep web servers oblivious to CT. Google Chrome and Apple's Safari require SCTs before accepting a certificate as valid, and steps towards further SCT verification have been taken recently [41]. Tor Browser does not require CT yet [10].

3 SAUTÉ ONIONS UNTIL DISCOVERY IS TRANSPARENT AND CONFECTION IS FIRM

3.1 System Goals

Let an onion association be unidirectional from a traditional domain name to an onion address. Three main system goals are as follows:

Privacy-Preserving Onion Associations Users should discover the same onion associations, and otherwise the possibility of an inconsistency must become public knowledge.

Forward Censorship Resistance Unavailability of a TLS site must not impede discovery of past onion associations.

Automated Verifiable Discovery Onion association search should be possible without requiring blind trust in third-parties. It must be hard to fabricate non-empty answers, and easy to automate the setup for scalability and robustness.

For comparison, today's onion location [36] does not assure a user that the same HTTP header is set for them as for everyone else. Classes of users that connect to a domain at different times or via different links can be given targeted redirects to distinct onion addresses without detection [44]. Onion location also does not work if a regular site becomes unavailable due to censorship. The *search engine approach* is further a frequent ask by Tor users [49]. The solutions that exist in practice rely on manually curated lists [29, 37, 47], notably with little or no retroactive accountability. As specified above, we aim for a similar utility but with a setup that can be automated for all onion associations and without the ability to easily fabricate non-empty answers without trivial detection. We sketch out how these security properties are achieved in Section 3.3.4.

3.2 Threat Model and Scope

We consider an attacker that wants to trick a user into visiting a targeted onionsite without anyone noticing the possibility of such behavior. Users are assumed to know the right traditional domain name that is easy to remember (such as `torproject.org`), but not its corresponding onion address. We further assume that the attacker either controls a trusted CA sufficiently to issue certificates or is able to deceive them sufficiently during certificate issuance to obtain a valid certificate from that CA. Any misbehavior is however assumed to be detectable in CT. So, the certificate ecosystem is treated as a *building block* that we make no attempt to improve.

We permit the attacker to make TLS sites unavailable after setup, but we assume it is difficult to censor the CT log ecosystem because it can be mirrored by anyone. Also, as part of the internet authentication infrastructure, adversaries may have equities conflicts in blocking CT logs, and if concerned at all about appearance would have a harder time justifying such a block versus, e.g., a political, journalism, or social media site. Similar to CT, we do not attempt to solve certificate revocation and especially not in relation to certificates that are connected to discovery of onion associations. This is consistent with Tor Browser's existing model for revocation with onion location, which similarly depends on the certificate for the redirecting domain. There is no formal counterpart to revoke a result in a search engine, but we outline future work related to this.

Our threat model includes countries that block direct access to HTTPS sites [26]. This is arguably a capable attacker, as no country is currently known to completely block indirect access via the Tor network (though in some places Tor bridges and/or obfuscated transport is needed). Our threat model also considers the plethora of blindly trusted parties that help users discover onion addresses with little or no retroactive accountability [1, 29, 37, 47]. In other words, it is in-scope to pave the path towards more accountability.

3.3 Description of Sauteed Onions

An observation that inspired work on sauteed onions is that onion location requires HTTPS [36]. This means that discovery of onion associations *already* relies on the CA ecosystem. By incorporating the use of CT, it is possible to add accountability to CAs and other parties that help with onion address discovery while also raising the bar for censoring sites and reducing anonymity. The name sauteed onions is a cooking pun; the association of an onion address with a domain name becomes transparent for everyone to see in CT logs.

For background, a CA-issued certificate can contain both a traditional domain name and a `.onion address` [5, 6]. This can be viewed as a mutual association because the issuing CA must verify the traditional domain name *and* the specified onion address. An immediate problem is that this would be ambiguous if there are multiple domain names; which one (if any) should be associated with an onion address with such certificate coalescence? A more appropriate path forward would therefore be to define an X.509v3 extension for sauteed onions which clearly *declares that a domain-validated name wants to be associated with an onion address.*

We describe two uses of sauteed onions that achieve our goals; first assuming it is easy to get CA-issued certificates that contain associated onion addresses for domain-validated names, and then a short-term roll-out approach that could make it a reality now. A sauteed onion is simply a CT-logged certificate that claims `example.com` wants to be associated with `<addr>.onion` but not necessarily the other way around, i.e., a unidirectional association.

3.3.1 Onion Location. Figure 1 illustrates onion location that uses certificates. A user establishes a TLS connection to a site as usual. Upon encountering a certificate that is CT-logged with an associated onion address for the visited site `example.com`, an onion-location prompt becomes available in Tor Browser or the onion site is visited automatically. This is the same type of redirect behavior as today's onion location [36], except that the possibility of such a redirect is disclosed in public CT logs. Attempts at targeted redirects would thus be visible to site owners and independent third-parties. A redirect to someone else's onion address would also be visible to the respective site owners. Notably the ability to detect inappropriate redirects acts as a deterrence while also being the first step towards remediation, e.g., if users bookmarked onion addresses [49] to achieve trust on first use or to avoid visiting a regular site *and* an onionsite in a way that might reduce a user's anonymity set.

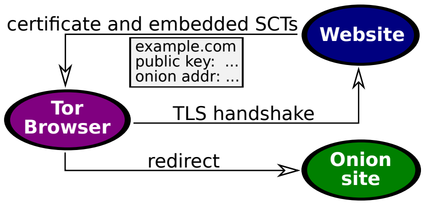

Figure 1: Onion location based on a CT-logged certificate.

A key observation is that onion location has always been a feature facilitated by TLS. By implementing it in certificates rather than HTTP headers that are delivered via HTTPS connections, TLS applications that are "not web" can use it too without rolling their own mechanisms. The addition of requiring CT to follow onion-location redirects is also an improvement compared to today, although one that could be achieved with an HTTP-based approach as well (or more ambitiously, for all Tor Browser certificate validations [10]).

We prototyped the above in a web extension that is free and open source [2]. The criterion for CT logging is at least one embedded SCT from a log in the policy used by Google Chrome [18]. If an onion-location redirect is followed, the path of the current webpage is preserved, similar to a typical configuration of today's HTTP-based onion location header that instead lists a complete URL [36].

3.3.2 Search Engine. A significant challenge for third-parties that help users discover TLS sites that are available as onion services is to gain confidence in the underlying dataset at scale. For example, SecureDrop onion names are scoped to news sites [47]; the list by Muffett is scoped as "no sites for tech with less than (arbitrary) 10,000 users" [29]; and `ahmia.fi` does not even attempt to give onion addresses human-meaningful names [34]. To make matters worse, solutions based on manually curated lists and third-party search are currently implemented with little or no accountability.

Figure 2 shows what our approach brings to the table. All CT logs can be monitored by a third-party to discover sauteed onions. A search API can then be presented to users for the resulting dataset, similar to existing monitoring services but scoped specifically for discovery of onion associations. The utility of such a search API is: "*what onion addresses are available for* `www.example.com`".

The expected behavior of the search API is that an answer can not be fabricated without controlling a CA or hijacking certificate

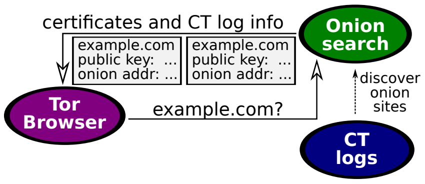

Figure 2: Verifiable domain name to onion address search.

issuance, and any CA malfeasance should further be caught by CT. This means that no party can fabricate inappropriate answers without detection. This is a major improvement compared to the alternative of no verifiability at all, although one that in and of itself does not prevent *false negatives*. In other words, available answers could trivially be omitted. This is a limitation with the authenticated data structure in CT that can be fixed; see security sketch in Section 3.3.4 for an intuition of how to work around it.

We specified an HTTP REST API that facilitates search using a domain name; the API also makes available additional information like the actual certificate and its exact index in a CT log. In total there are two endpoints: `search` (list of matches with identifiers to more info) and `get` (more info). The complete API specification is available online together with our implementation, which is free and open source [2]. An independent implementation from Tor's hack week is also available by Rhatto [38]. Our prototype runs against all CT logs in Google Chrome for certificates logged after July 16, 02022. A few query examples are available in Appendix A.

3.3.3 Certificate Format. Until now we assumed that a sauteed onion is easily set up, e.g., using an X.509v3 extension. The bad news is that such an extension does not exist, and it would likely be a long journey to standardize and see deployment by CAs. Therefore, our prototypes rely on a backwards-compatible approach that encodes onion addresses as subdomains [43]. To declare that `example.com` wants to be associated with `<addr>.onion`, one can request a domain-validated certificate that contains both `example.com` and `<addr>onion.example.com` [45]. The inclusion of `example.com` ensures that such a setup does not result in a dangerous label [17]. The *hack to encode an onion address as a subdomain* makes it part of the certificate without requiring changes to CAs. Appendix B details the necessary setup-steps further. The gist is the addition of a subdomain DNS record and using the `-d` option in `certbot` [15].

Although the subdomain approach is easy to deploy right now, it is by no means a perfect solution. An X.509v3 extension would not require the configuration of an additional DNS record. In other words, the unidirectional sauteed onions property works just as well if the subdomain is not domain-validated. The important part is that the CA validates `example.com`, and that the associated onion address can be declared somewhere in the issued certificate without an ambiguous intent. Another imperfection that goes hand-in-hand with backwards-compatibility is that CAs would have to *opt-out* from sauteed onions, unlike site owners that instead have to *opt-in*.

To avoid recommending a pattern that is discouraged by CAs, the Tor Project should at least have a dialog with Let's Encrypt which issues the most certificates [3]. Somewhat similar subdomain hacks related to CAs exist, but then with explicit negotiations [46]. Subdomain hacks without a relation to CAs and TLS were discouraged in the past [22]. We argue that sauteed onions is related because CA-validated names are at the heart of our approach. For example, this is unlike Mozilla's binary transparency idea that just wanted to reuse a public log [28]. Sauteed onions also do not result in more issued certificates; it is just the number of domain-validated names that increase by one for TLS sites that do the setup.

3.3.4 Security sketch. Our threat model disallows the attacker to tamper with CT and to make the log ecosystem unavailable. Onion location as described in Section 3.3.1 therefore ensures that a redirect becomes public, achieving detectability as defined in our privacy-preserving onion association goal. The search engine in Section 3.3.2 trivially achieves the same goal because onion associations are *found* via CT. Blocking a TLS site is additionally *too late* if an association is already in a CT log, thus achieving forward censorship resistance. Our search engine approach further makes it hard to forge non-answers without detection because it requires control of a CA and defeating the tamper-evidence of CT logs. While it is possible to omit available answers, this can be mitigated by having multiple search APIs, domains that check the integrity of their own onion associations similar to the proposed verification pattern in CONIKS [25], or to represent the sauteed onion dataset as a sparse Merkle tree to get a verifiable log-backed map that additionally supports efficient non-membership proofs that CT lacks [9, 14].

3.4 Future Work

It would be valuable to implement proofs of no omissions as well as native lookups in a web extension or Tor Browser to verify everything before showing the user a result (certificates, proofs of logging, etc). The entire or selected parts of the sauteed onion dataset may further be delivered to Tor Browser similar to SecureDrop onion names [47]. The difference would be that the list is automated using a selection criteria from CT logs rather than doing it manually on a case-by-case basis. A major benefit is that the sauteed onion dataset can then be queried locally, completely avoiding third-party queries and visits to the regular site. Another approach to explore is potential integration of the sauteed onion dataset into Tor's DHT: a cryptographic source of truth for available onion associations is likely a helpful starting point so that there is *something to distribute*. It would also be interesting to consider other search-engine policies than *show everything* as in our work, e.g., only first association or last association. (These policies can be verified with *full audits* [14].)

4 RELATED WORK

The CA/B forum accepts certificates with `.onion` addresses [5, 6]. DigiCert supports extended validation of `.onion` addresses [11], and HARICA domain validation [19]. Muffett proposed same-origin onion certificates that permit clients to omit verification of the CA trust chain for onionsites [30]. Sauteed onions help Tor users *discover* domain names with associated onion addresses. Therefore, it is a complement to approaches that bring HTTPS to onionsites.

Syverson suggested that traditional domain names and `.onion` addresses can be glued into a single registered domain [43]. Nusenu proposed long-term Tor relay identifiers based on domain names to retrieve lists of relevant public keys via HTTPS [35]. Sauteed onions may be used for such associations with the benefit of transparency, and it is further a *lighter* version of Syverson and Traudt's self-authenticated traditional addresses which favors early deployment over properties like bidirectional onion associations, guaranteed timeliness of revocation, and addressing all known threats [44, 45].

Winter *et al.* studied how users engage with onion services [49]. A gist is that Tor users have a hard time discovering onion addresses and verifying them as authentic. Common discovery mechanisms that are associated with human-meaningful identifiers include personal communication, webpage links, onion-location redirects [36], third-party lists [37], and search engines like DuckDuckGo. Prior work has also focused on enumerating onion addresses without any associated identity, e.g., through CT-logged certificates with `.onion` addresses [29] and crawling [1, 34]. Sauteed onions enhance onion location by making the claimed associations transparent in CT, and facilitate third-party solutions with less blind trust and without assumptions about TLS sites not becoming blocked in the future.

Several ideas were proposed that mitigate or bypass the problem of random-looking onion addresses. Some sites generate vanity addresses that, e.g., start with a prefix and have other memorable traits [27]. Fink sketched out how to map onion addresses to a set of words [16]. Kadianakis *et al.* defined a common API to hook into alternative naming systems that give onion addresses pet names [20]. SecureDrop Onion Names is one such example that is, however, implemented directly in Tor Browser as an HTTPS Everywhere ruleset for selected news sites. Other alternative naming systems include Namecoin [31] and OnioNS [48]. Sauteed onions is also an alternative naming system, but one that relies on CAs and CT logs. It may be possible to construct sauteed onions via DNSSEC, but then relying on the DNS hierarchy without transparency logging. Scaife *et al.* [40] proposed the `.o` TLD as an onionsite with DNSSEC.

Nordberg connected transparency logs and the consensus mechanism that Tor uses [32]. Dahlberg *et al.* proposed CT in Tor for all certificate validations [10]. We only check signatures of embedded SCTs in relation to onion location, and our search engine is a simple application of CT monitoring. There is a large body of orthogonal work that improve CAs and CT. For example, multi-path domain-validation makes it harder to hijack onion associations [4], and deployment of gossip would harden our CT log assumptions [7, 33].

5 CONCLUSION

Sauteed onions declare unidirectional associations from domain names to onion addresses. These onion associations are established in CA-issued and CT-logged TLS certificates, thereby making them public for everyone to see. We propose two immediate applications: certificate-based onion location and more automated verifiable search. Both applications are opt-in for domain owners, and rely on similar assumptions as today's onion location. The added benefit is more transparency, which facilitates a higher degree of consistency between found onion associations as well as more censorship-resistance for TLS sites after setup. Configuration of sauteed onions requires one more DNS record and a domain-validated certificate from any CA (such as Let's Encrypt). In the future, the additional DNS record may be replaced by an X.509v3 extension. We leave it as a fun exercise to find the onion address of a TLS site that is intentionally being censored by us: `blocked.sauteed-onions.org`.

ACKNOWLEDGMENTS

We would like to thank Kushal Das, Daniel Kahn Gillmor, Silvio Rhatto, and Tobias Pulls for helpful discussions and comments. Rasmus Dahlberg was supported by the Swedish Foundation for Strategic Research. Paul Syverson was supported by ONR.

REFERENCES

[1] Ahmia. Indexing and crawling. https://ahmia.fi/documentation/indexing/.
[2] Artifact. https://gitlab.torproject.org/tpo/onion-services/sauteed-onions, 02022.
[3] J. Aas, R. Barnes, B. Case, Z. Durumeric, P. Eckersley, A. Flores-López, J.A. Halderman, J. Hoffman-Andrews, J. Kasten, E. Rescorla, S.D. Schoen, B. Warren. Let's Encrypt: An automated certificate authority to encrypt the entire web. *Proc. ACM CCS*, 02019.
[4] H. Birge-Lee, L. Wang, D. McCarney, R. Shoemaker, J. Rexford, P. Mittal. Experiences deploying multi-vantage-point domain validation at Let's Encrypt. *Proc. USENIX Sec.*, 02021.
[5] CA/Browser Forum. Ballot 144 – validation rules for .onion names. https://cabforum.org/2015/02/18/ballot-144-validation-rules-dot-onion-names/, 02015.
[6] CA/Browser Forum. Ballot sc27v3: Version 3 onion certificates. https://cabforum.org/2020/02/20/ballot-sc27v3-version-3-onion-certificates/, 02020.
[7] L. Chuat, P. Szalachowski, A. Perrig, B. Laurie, E. Messeri. Efficient gossip protocols for verifying the consistency of certificate logs. *Proc. IEEE CNS*, 02015.
[8] R. Dahlberg T. Pulls. Verifiable light-weight monitoring for Certificate Transparency logs. *Proc. NordSec*, 02018.
[9] R. Dahlberg, T. Pulls, R. Peeters. Efficient sparse Merkle trees - caching strategies and secure (non-)membership proofs. *Proc. NordSec*, 02016.
[10] R. Dahlberg, T. Pulls, T. Ritter, P. Syverson. Privacy-preserving & incrementally-deployable support for Certificate Transparency in Tor. *PoPETs* 2021(2), 02021.
[11] DigiCert Inc. Ordering a .onion certificate from DigiCert. https://www.digicert.com/blog/ordering-a-onion-certificate-from-digicert.
[12] R. Dingledine, N. Mathewson, P. Syverson. Tor: The Second-Generation Onion Router. *Proc. USENIX Sec.*, 02004.
[13] B. Dowling, F. Günther, U. Herath, D. Stebila. Secure logging schemes and certificate transparency. *Proc. ESORICS*, 02016.
[14] A. Eijdenberg, B. Laurie, A. Cutter. Verifiable data structures. https://github.com/google/trillian/blob/111e9369ab032e493a2f19f9be6d16c4f78ccca5/docs/papers/VerifiableDataStructures.pdf, 02018.
[15] EFF. Changing a certificate's domain. https://eff-certbot.readthedocs.io/en/stable/using.html#changing-a-certificate-s-domains.
[16] A. Fink. Mnemonic .onion URLs. https://gitweb.torproject.org/torspec.git/tree/proposals/194-mnemonic-urls.txt, 02012.
[17] D.K. Gillmor. Dangerous Labels in DNS and E-mail. Internet-Draft draft-dkg-intarea-dangerous-labels-01, IETF, 02022.
[18] Google Inc. Known logs. https://github.com/google/certificate-transparency-community-site/blob/master/docs/google/known-logs.md, 02022.
[19] Harica. DV certificates for onion websites. https://news.harica.gr/article/onion_announcement/, 02021.
[20] G. Kadianakis, Y. Angel, D. Goulet. A name system API for Tor Onion Services. https://gitweb.torproject.org/torspec.git/tree/proposals/279-naming-layer-api.txt, 02016.
[21] B. Laurie. Certificate Transparency. *Commun. ACM*, 57(10):40–46, 02014.
[22] B. Laurie. Re: [Trans] Mozilla's basic take on binary transparency. https://mailarchive.ietf.org/arch/msg/trans/1FxzTkn4LVxU6KN2P3YfbVsKpho/, 02017.
[23] B. Laurie, A. Langley, E. Kasper. Certificate Transparency. https://tools.ietf.org/html/rfc6962, 02013.
[24] B. Li, J. Lin, F. Li, Q. Wang, Qi Li, J. Jing, C. Wang. Certificate Transparency in the wild: Exploring the reliability of monitors. *Proc. ACM CCS*, 02019.
[25] M.S. Melara, A. Blankstein, J. Bonneau, E.W. Felten, M.J. Freedman. CONIKS: Bringing key transparency to end users. *USENIX Sec.*, 02015.
[26] S. Migliano S. Woodhams. Websites blocked in Russia since Ukraine invasion. https://www.top10vpn.com/research/websites-blocked-in-russia/, 02022.
[27] mkp224o—vanity address generator for ed25519 onion services. https://github.com/cathugger/mkp224o, 02022.
[28] Mozilla. Security/binary transparency. https://wiki.mozilla.org/Security/Binary_Transparency, 02017.
[29] A. Muffet. Real-world onion sites. https://github.com/alecmuffett/real-world-onion-sites, 02022.
[30] A. Muffett. Same origin onion certificates. https://crt.sh/?id=6819596552, 02020.
[31] Namecoin. https://www.namecoin.org/.
[32] L. Nordberg. Tor consensus transparency. https://gitweb.torproject.org/torspec.git/tree/proposals/267-tor-consensus-transparency.txt, 02014.
[33] L. Nordberg, D.K. Gillmor, T. Ritter. Gossiping in CT. Internet-draft draft-ietf-trans-gossip-05, IETF, 02018.
[34] J. Nurmi. *Understanding the Usage of Anonymous Onion Services.* PhD thesis, Tampere University, Tampere, Finland, 02019.
[35] nusenu. HAROI: Human readable authenticated relay operator identifier. https://lists.torproject.org/pipermail/tor-dev/2021-December/014688.html, 02021.
[36] Tor Project. Onion-location. https://community.torproject.org/onion-services/advanced/onion-location/.
[37] Tor Project. Onion services. https://community.torproject.org/onion-services/.
[38] Silvio Rhatto. Sauteed week API backend. https://gitlab.torproject.org/rhatto/sauteed-week/-/blob/main/docs/api.md, 02022.
[39] Sauteed onion certificate. https://crt.sh/?id=5957691193, 02022.
[40] N. Scaife, H. Carter, P. Traynor. OnionDNS: A seizure-resistant top-level domain. *Proc. IEEE CNS*, 02015.
[41] E. Stark, J. DeBlasio, D. O'Brien, D. Balzarotti, W. Enck, S. King, A. Stavrou. Certificate transparency in google chrome: Past, present, and future. *IEEE Secur. Priv.*, 19(6):112–118, 02021.
[42] E. Stark, R. Sleevi, R. Muminovic, D. O'Brien, E. Messeri, A. Porter Felt, B. McMillion, P. Tabriz. Does certificate transparency break the web? measuring adoption and error rate. *Proc. IEEE Security and Privacy*, 02019.
[43] P. Syverson. The once and future onion. *Proc. ESORICS*, 02017.
[44] P. Syverson, M. Finkel, S. Eskandarian, D. Boneh. Attacks on onion discovery and remedies via self-authenticating traditional addresses. *ACM WPES*, 02021.
[45] P. Syverson M. Traudt. Self-authenticating traditional domain names. *Proc. IEEE SecDev*, 02019.
[46] F. Valsorda. How Plex is doing HTTPS for all its users. https://words.filippo.io/how-plex-is-doing-https-for-all-its-users/, 02015.
[47] SecureDrop. Getting an onion name for your SecureDrop. https://securedrop.org/faq/getting-onion-name-your-securedrop/.
[48] J. Victors, M. Li, X. Fu. The onion name system. *PoPETs*, 02017(1), 02017.
[49] P. Winter, A. Edmundson, L.M. Roberts, A. Dutkowska-Zuk, M. Chetty, N. Feamster. How do Tor users interact with onion services? *Proc.USENIX Sec.*, 02018.

A ONION ASSOCIATION SEARCH EXAMPLES

We host the search engine described in Section 3.3.2 on a Debian VM with 1GB RAM, 20GB SSD, and a single vCPU. It is available at `api.sauteed-onions.org` as well as `zpadxxmoi42k45iifrzuktwqktihf5didbaec3xo4dhvlw2hj54doiqd.onion`. Please note that we operate this prototype on a best-effort level until December, 02022.

An example for the `search` endpoint is provided in Figure 3, followed by extracting additional certificate information using the `get` endpoint in Figure 4. There are many CT-logged certificates for the same onion association because certificates are renewed periodically and typically submitted to multiple CT logs.

B CONFIGURATION EXAMPLE

We used `certbot` to set up sauteed onions using Let's Encrypt and `apache` on a Debian system. The difference when compared to the usual `certbot` instructions is that the `-d` flag must be specified to enumerate all SANs as a comma-separated list [15]. The domain name with an associated onion address as a subdomain also needs to be reachable via DNS for Let's Encrypt to perform domain validation. Therefore, an appropriate A/AAAA or CNAME record is required. A sanity-check for `www.sauteed-onions.org` would be to verify that `dig qvrbktnwsztjnbga6yyjbwzsdjw7u5a6vsyzv6hkj75clog4pdvy4cydonion.www.sauteed-onions.org` returns the same IP address as `dig www.sauteed-onions.org` before running `certbot -apache -d www.sauteed-onions.org,qvrbktnwsztjnbga6yyjbwzsdjw7u5a6vsyzv6hkj75clog4pdvy4cydonion.www.sauteed-onions.org`. See `crt.sh` for an example certificate [39].

```
$ curl -s https://api.sauteed-onions.org/search?in=www.sauteed-onions.org | json_pp
[
   {
      "identifiers" : [
         "2",
         "3",
         "24",
         "25",
         "28",
         "29",
         "37"
      ],
      "onion_addr" : "qvrbktnwsztjnbga6yyjbwzsdjw7u5a6vsyzv6hkj75clog4pdvy4cyd.onion",
      "domain_name" : "www.sauteed-onions.org"
   }
]
```

Figure 3: Find onion associations for `www.sauteed-onions.org`.

```
$ curl -s https://api.sauteed-onions.org/get?id=2 | json_pp
{
   "onion_addr" : "qvrbktnwsztjnbga6yyjbwzsdjw7u5a6vsyzv6hkj75clog4pdvy4cyd.onion",
   "domain_name" : "www.sauteed-onions.org",
   "log_id" : "b1N2rDHwMRnYmQCkURX/dxUcEdkCwQApBo2yCJo32RM=",
   "log_index" : 582362461,
   "cert_path" : "db/logs/Mammoth/582362461.pem"
}
$ curl -L https://api.sauteed-onions.org/db/logs/Mammoth/582362461.pem | \
  openssl x509 -text -noout
...
```

Figure 4: Get further information relating to the certificate with identifier "2".